Atty. Docket No. 42390P11446
Express Mail Label No. EL802873966US

UNITED STATES PATENT APPLICATION

FOR

Hardware Updated Metadata For Non-Volatile Mass Storage Cache

INVENTOR:
Richard L. Coulson

PREPARED BY:
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025
(714) 557-3800

10130830 . 09254690

Hardware Updated Metadata for Non-Volatile Mass Storage Cache

Field

The present invention relates in general to cache memories, and in particular, to a method and apparatus for providing a mass storage cache management.

General Background

09452651 .083101

The use of a cache with a processor reduces memory access time and increases the overall speed of a device. Typically, a cache is an area of memory which serves as a temporary storage area for a device. Data frequently accessed by the processor remain in the cache after an initial access and subsequent accesses to the same data may be made to the cache.

When the processor refers to a memory and finds the data in the cache, it is said to produce a hit. Otherwise, when the data is not in the cache but in the memory, a miss occurs. If a miss occurs, an allocation may be made to place the data into the cache for later accesses. Here, an access can be for loading data to the processor or for storing data from the processor to a memory. The cached information is retained by the cache until it is no longer needed, made invalid or replaced by other data, in which instances the cache entry is de-allocated.

In particular, for mass storage devices, the size of a memory is significantly larger than the amount of space available in a cache so that the cache eventually becomes full. When the cache is full, it is necessary to replace or de-allocate existing lines of stored data in the cache memory to make room for newly requested lines of data.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described in detail with reference to the following drawings in which like reference numerals refer to like elements wherein:

Figure 1 shows an exemplary system implementing the invention;

Figure 2 is an exemplary cache line layout in accordance to one embodiment of the invention; and

Figure 3 is an exemplary block diagram of one embodiment of the invention.

09945266.083101

DETAILED DESCRIPTION

In the following description, specific details are given to provide a thorough understanding of the invention. For example, some circuits are shown in block diagram in order not to obscure the present invention in unnecessary detail. However, it will be appreciated by those skilled in the art that the present invention may be practiced without such specific details.

As disclosed herein, a "cache" refers to a temporary storage area and can be any combination of a volatile storage such as a random access memory (RAM) and a nonvolatile storage. The term "machine readable medium" includes, but is not limited to portable or fixed storage devices, optical storage devices, and any other devices capable of storing instructions and/or data. The term "instructions" refers to a single instruction or a group of instructions, and may be one or a combination of software and firmware including data, codes, and programs that can be read and/or executed to perform certain tasks.

10783056.091501

One embodiment of the invention provides an apparatus and method to manage cache memories. For example, two pieces of metadata for mass storage caches are typically used to decide what to de-allocate/replace in a cache when the cache is considered full. The first is a count of the number of hits to a cache line and the second is an indication of the last time the cache line was accessed, for example the least recently used (LRU) information. When the cache is accessed, the metadata should be accurately updated for use in the de-allocation/replacement decisions.

Also, in one embodiment, a clock with a given number of periods is used to approximate usage information, such as the LRU information, for a cache. Generally, a current clock period is checked when a cache is accessed and a usage bit corresponding to the current clock period is set to indicate the usage information for the cache. Based on the usage information, de-allocation or replacement decisions are made to remove data from the cache when the cache storage becomes too large or is considered full, for example, nearly full or full. When available, a "writeback" cycle is used in updating the metadata.

An exemplary embodiment of a system 100 implementing the principles of the invention is shown in Figure 1. The system 100 includes an input/output (I/O) control hub 110 which is coupled to a memory control hub 120 and a cache controller 130. The memory control hub 120 is coupled to a processor 140 and a main memory 150. The cache controller 130 is coupled to a storage cache memory (hereinafter "cache") 160 and a mass storage device 170. Finally, a number of input/output devices 180 such as a keyboard, mouse and/or display can also be coupled to the I/O control hub 110.

In the system 100, the memory control hub 120 and the I/O control hub 110 may be implemented in a chipset. The main memory 150 may comprise of a random-access-memory (RAM). The mass storage device 170 may be, for example, a magnetic or optical memory device, for mass storage (or saving) of information. Also, in one embodiment, the cache memory 160 is a non-volatile memory such as a polymer ferroelectric RAM (PFRAM). The cache controller 130 including a least recently used (LRU) logic 135 controls the operation of the cache 160, such as the metadata updates and cache de-allocation/replacement.

09955256.083101

Although the system 100 is shown as system with a single processor, the invention may be implemented with multiple processors. In such case, each additional processor would share the cache 160 and main memory 100 for writing data and/or instructions to and reading data and/or instructions from the same. Similarly, the invention may be implemented with multiple mass storage devices, each having a cache or sharing a cache. Moreover, in an alternative embodiment of the system 100, the cache may be controlled by a driver and executed on the processor through the I/O control hub 110. The driver may be stored in the main memory 150 and may access data from the mass storage 170. When appropriate, the driver stores the accessed data in the cache 160. In such case, the cache controller 130 can be omitted and the LRU logic would be an algorithm in the driver. Also, the cache 160 and mass storage 170 would be directly coupled to the I/O control hub 110.

09645256 .083101

In addition, an external clock may be coupled to the system 100, based on which the processor 140 controls the operation of the cache controller 130. In one embodiment, the clock may be implemented as an internal clock, either by hardware or software, within the cache controller 130. The invention will next be described below.

Figure 2 shows an example cache line layout for 512 bytes of data 210 with metadata 220. The metadata 220 may include, as shown, an error correction code (ECC) 230 to recover the data 210, a tag or address 240 to indicate the data to which the metadata corresponds, and flags 250 to indicate if the cache line is valid and/or dirty. The metadata 220 also may include a hit count 260 and the usage timeframe indicator 270, used to implement the invention.

The hit count 260 is updated when the cache line is read or written and the tag/address is unchanged. The hit count 260 can be updated by incrementing the hit count using a hit counter. In one embodiment, the hit counter increments (decrements) until a maximum (minimum) value is reached. For instance, 4-bit counter having a maximum value of 15 would remain at 15 once this maximum value is reached.

The usage timeframe information 270 approximates the usage information such as the LRU information used in deciding the de-allocation/replacement of the data in the cache. To implement true least recently used indications in hardware is time consuming and space intensive in software. Therefore, an approximation is developed that can easily be implemented in hardware. Namely, a clock having N periods is utilized. Each clock period is fairly long and the exact value is something that should be tuned to the workload being cached. For very large caches, the clock "tick" (the time taken by the clock to pass from one period to the next) can be in hours or even tens of hours. The clock cycle resets after the Nth period.

For example, the cache line layout in Figure 2 shows a usage timeframe 270 for N = 4. In such case, four usage bits are allocated to represent the usage timeframe information 270, each corresponding to one of the clock periods 0-3. The clock period would reset to 0 after the 3rd period and the clock output would be 0,1,2,3,0,1,2,3...

Figure 3 shows an exemplary embodiment of a procedure 300 to update the usage timeframe information in accordance with the invention. When a cache line is accessed (block 310) for either a read or a write, the current clock period of the clock 180 is checked (block 320). The usage bit corresponding to the current clock period is then set (block 330). If the usage bit has already been set, it remains set. In one embodiment, if the cache line is accessed just to read the metadata, the usage bit is not changed.

For example, in the usage timeframe information 270 of Figure 2, if the clock period is 0 when cache line is accessed, the period 0 usage bit is set. If the period 0 usage bit is already set, then it remains set. If the period is 4, the usage bit corresponding to period 4 is set.

Also, in one embodiment of the procedure described above, the usage bit is set or written during the writeback cycle. During a writeback cycle, data read from a memory is rewritten back because the data is destroyed in the reading process. Examples of memories with a writeback cycle (hereinafter "destructive read" memory) includes, but is not limited to, a memory with a destructive read cycle such as the PFRAM, a Magnetic RAM (MRAM) and a core memory. Accordingly, if a destructive read memory is used as the cache memory 160, metadata information including the usage bit is updated using the writeback cycle.

Referring back to Figure 3, when a new clock period begins (block 340), the usage bits corresponding to the new clock period are erased (block 350). Thus, when a clock ticks from one period to the next, all of the usage bits for that time period are erased. For a destructive read cache memory, the usage bits can be erased, for example, by reading them. Also, in one embodiment, the feature of the memory array is used to erase the group of bits at once.

Also, if an address/tag 240 in the metadata is changed, a new cache line corresponding to a new address is generated. Thus, when a tag address is changed (block 360), the usage bits for the new address are reset (block 370).

09945256.083101

Thereafter, the current clock period is check and the usage bit for the current period is set (blocks 320 and 330).

The result is that for every cache line, there is an accurate representation of whether or not that cache line has been accessed in the last N number of periods. In Figure 2, for example, if the current clock period is 3 while the usage bits for periods 0, 1, 2 and 3 are all set, the cache line has been used in each of the previous 3 and current clock period. Therefore, this cache line is likely a poor candidate for replacement. However, a cache line with none of the usage bits set, meaning no accesses during the last 3 plus current time period, is likely a good candidate for replacement. In another example, a cache line with only the usage bit corresponding to the oldest periods being set indicates that it's been a long time since the cache line has been accessed. Such cache line might also be an attractive replacement candidate.

Accordingly, the invention allows reliable data on which to base allocation or replacement and other decisions requiring the hit count and/or LRU information. For example, when a cache storage is considered full, decisions to de-allocate/replace data in the cache can be made based on both the hit count and the usage timeframe or based solely on the usage timeframe. By using the usage timeframe to decide what to de-allocate/replace when a cache is full, neither a software LRU list nor the hit count needs to be maintained. This raises the performance as fewer instructions need to be executed. Also, by using a non-volatile memory such as the PFRAM, costs are significantly reduced as list and counter in software use Dynamic Random Access Memory (DRAM) space, which is more expensive than a non-volatile memory.

Moreover, in a destructive read memory, the hit count and usage timeframe information can be automatically updated without software intervention, although the results are reported to software. Also, the updates can be performed without any performance overhead as a writeback cycle is required and simple updates of the metadata, including the usage bits, can be written during the writeback cycle. Therefore, the data required for de-allocation/replacement decisions can be provided without requiring any extra accesses to the memory array since the updates are accomplished in the writeback cycle.

While the invention has been described with reference to a cache memory, the teachings of the invention can be applied to any types of memory with metadata. Also, the invention has been described with reference to metadata indicating the usage information such as the least recently used information, the invention can be applied to other types of metadata. For example, the writeback cycle can be used to update any metadata information for a destructive read memory, either volatile or non-volatile. Therefore, the foregoing embodiments are merely exemplary and are not to be construed as limiting the present invention. The present teachings can be readily applied to other types of apparatuses. The description of the present invention is intended to be illustrative, and not to limit the scope of the claims. Many alternatives, modifications, and variations will be apparent to those skilled in the art.

10032763.095450